

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号  
特許第4409427号  
(P4409427)

(45) 発行日 平成22年2月3日 (2010.2.3)

(24) 登録日 平成21年11月20日 (2009.11.20)

(51) Int.Cl.  
G 0 6 F 9 / 4 8 (2006.01)

F I  
G O 6 F 9 / 4 6 4 5 5 C

請求項の数 7 (全 14 頁)

(21) 出願番号	特願2004-509742 (P2004-509742)	(73) 特許権者	504199127
(86) (22) 出願日	平成15年5月7日 (2003.5.7)		フリースケール セミコンダクター イン
(65) 公表番号	特表2006-502470 (P2006-502470A)		コーポレイテッド
(43) 公表日	平成18年1月19日 (2006.1.19)		アメリカ合衆国 7 8 7 3 5 テキサス州
(86) 国際出願番号	PCT/US2003/014215		オースティン ウィリアム キャノン
(87) 国際公開番号	W02003/102723		ドライブ ウェスト 6 5 0 1
(87) 国際公開日	平成15年12月11日 (2003.12.11)	(74) 代理人	100142907
審査請求日	平成18年4月26日 (2006.4.26)		弁理士 本田 淳
(31) 優先権主張番号	10/159,386	(74) 代理人	100149641
(32) 優先日	平成14年5月31日 (2002.5.31)		弁理士 池上 美穂
(33) 優先権主張国	米国 (US)	(74) 代理人	100116322
			弁理士 桑垣 衛

最終頁に続く

(54) 【発明の名称】 複数のレジスタ・コンテキストを有するデータ処理システムおよび該システムのための方法

(57) 【特許請求の範囲】

【請求項 1】

レジスタファイル ( 3 2 ) を有するプロセッサ ( 1 2 ) を備えるデータ処理システム ( 1 0 ) であって、

前記レジスタファイル ( 3 2 ) は、割込動作毎、プロセス動作毎、およびスレッド動作毎のうちの少なくとも一つに応じてそれぞれ前記プロセッサ ( 1 2 ) によって切替えて用いられる複数のレジスタコンテキストを有し、

前記レジスタコンテキスト ( 7 4 ) には、当該レジスタコンテキスト ( 7 4 ) の汎用レジスタが他のレジスタコンテキスト ( 7 2 ) の汎用レジスタにマップ ( 8 2 ) されているか否かを示すコンテキスト制御レジスタ ( 7 5 ) がそれぞれ対応付けられ、

前記プロセッサ ( 1 2 ) がアクセスした前記汎用レジスタが他の汎用レジスタにマップ ( 8 2 ) されている場合、前記プロセッサ ( 1 2 ) は当該他の汎用レジスタにもアクセスすることを特徴とする、データ処理システム。

【請求項 2】

複数の前記コンテキスト制御レジスタ ( 7 5 , 7 7 , 7 9 ; 1 2 8 , 1 3 0 , 1 3 2 ) は、それぞれ前記レジスタコンテキスト ( 7 4 , 7 0 , 7 2 ; 9 4 , 9 0 , 9 2 ) に含まれる、請求項 1 記載のデータ処理システム。

【請求項 3】

複数の前記コンテキスト制御レジスタ ( 7 5 , 7 7 , 7 9 ; 1 2 8 , 1 3 0 , 1 3 2 ) は、前記レジスタファイル ( 3 2 ) が有する制御レジスタファイル ( 3 8 ) に含まれる、

10

20

請求項 1 記載のデータ処理システム。

【請求項 4】

前記コンテキスト制御レジスタ ( 1 4 0 ) は更に、  
スタックポインタのマッピングを有効にするスタックポインタコンテキストイネーブルフィールド ( 1 5 2 ) と；

前記スタックポインタに関するマップ先のレジスタコンテキストを選択するスタックポインタコンテキスト選択フィールド ( 1 5 4 ) と  
を有する、請求項 1 ~ 3 何れか一項記載のデータ処理システム。

【請求項 5】

前記コンテキスト制御レジスタ ( 1 4 0 ) は更に、  
アサートされることによって複数のレジスタコンテキストを有効にするイネーブルフィールド ( 1 4 2 ) を有する、請求項 1 ~ 4 何れか一項記載のデータ処理システム。

【請求項 6】

前記コンテキスト制御レジスタ ( 1 4 0 ) は更に、  
現行のレジスタコンテキストの前に実行されていたレジスタコンテキストを特定するための保存済コンテキストフィールド ( 1 4 8 ) を含む、請求項 1 ~ 5 何れか一項記載のデータ処理システム。

【請求項 7】

レジスタファイル ( 3 2 ) を有するプロセッサ ( 1 2 ) におけるデータ処理方法であって、

前記レジスタファイル ( 3 2 ) は、割り込動作毎、プロセス動作毎、およびスレッド動作毎のうちの少なくとも一つに応じてそれぞれ前記プロセッサ ( 1 2 ) によって切替えて用いられる複数のレジスタコンテキストを有し、

前記レジスタコンテキスト ( 7 4 ) には、当該レジスタコンテキスト ( 7 4 ) の汎用レジスタが他のレジスタコンテキスト ( 7 2 ) の汎用レジスタにマップ ( 8 2 ) されているか否かを示すコンテキスト制御レジスタ ( 7 5 ) がそれぞれ対応付けられ、

前記プロセッサ ( 1 2 ) がアクセスした前記汎用レジスタが他の汎用レジスタにマップ ( 8 2 ) されている場合、前記プロセッサ ( 1 2 ) は当該他の汎用レジスタにもアクセスすることを特徴とする、データ処理方法。

【発明の詳細な説明】

【技術分野】

【 0 0 0 1 】

本発明は、一般に、データ処理システムに関し、より詳細には、複数のレジスタ・コンテキストを有するデータ処理システムに関する。

【背景技術】

【 0 0 0 2 】

マイクロプロセッサなどのデータ処理システムでは、動作の実行および処理を制御するのにプロセッサが利用される。プロセッサは、通常の動作中および例外処理中にプロセッサによって利用されるレジスタ・コンテキストを格納するレジスタ群を含む。割り込み、またはプロセス切り替えが生じた場合、割り込み処理プログラムまたは新たなプロセスが同一のレジスタ群を使用し、レジスタ内の値の一部を変更する可能性があるため、レジスタ・コンテキスト情報が壊れる可能性がある。

【 0 0 0 3 】

上述の問題の 1 つの解決法は、割り込みの処理または新たなプロセスを開始するのに先立って、レジスタ・コンテキストの現在の値をメモリの中に保存し、割り込み処理が完了した時点、または現在のプロセスに戻る時点で、メモリからレジスタ群の中に保存されているレジスタ・コンテキスト値を再び読み込むことである。しかし、レジスタ・コンテキストを保存すること、および新たなコンテキストをロードすることのオーバーヘッドは、リアルタイム環境または高性能環境においては望ましくない。したがって、柔軟性があり、オーバーヘッドを減らすデータ処理システムにおけるレジスタ・コンテキスト選択方式

10

20

30

40

50

の必要性が存在する。

【発明を実施するための最良の形態】

【0004】

本発明を、例として、限定としてではなく、添付の図によって示す。図では、同様の符号は同様の要素を示す。

図における諸要素は、簡明にするために示しており、必ずしも原寸に比例して描かれているわけではないことは、当業者には理解されよう。例えば、図における諸要素の一部の寸法は、本発明の諸実施形態の理解を向上させるのに役立つよう、他の諸要素と比べて誇張されている可能性がある。

【0005】

本明細書で使用する「バス」という用語は、データ、アドレス、制御、またはステータスなどの1つまたは複数の様々なタイプの情報を転送するのに使用することが可能な複数の信号または導体を指すのに使用する。「アサートする」および「ネゲートする」（または「デアサートする」）という用語は、信号、ステータス・ビット、または類似の装置を論理的に真の状態、または論理的に偽の状態にすることを指す場合にそれぞれ使用される。論理的に真の状態が論理レベル1である場合、論理的に偽の状態は論理レベル0である。また、論理的に真の状態が論理レベル0である場合、論理的に偽の状態は論理レベル1である。数の前に付いた記号「\$」は、その数が16進形式で、つまり16を基数とする形式で表されていることを示す。数の前に付いた記号「%」は、その数が2進形式、つまり2を基数とする形式で表されていることを示す。

【0006】

図1は、データ・バス4およびアドレス・バス6を介して外部デバイス2に結合されたデータ処理システム10を示している。データ処理システム10は、プロセッサ12を含む。一実施形態では、データ処理システム10および外部デバイス2はそれぞれ、別々の集積回路として実装される。代替の実施形態では、データ処理システム10および外部デバイス2は、単一の集積回路上に実装することが可能である。データ処理システム10内部では、プロセッサ12が、内部データ・バス13および内部アドレス・バス14でシステム集積回路22に結合される。

【0007】

本発明の一部の実施形態では、データ処理システム10は、単一の集積回路上に形成されることに留意されたい。さらに、一部の実施形態では、データ処理システム10は、シングル・チップ・マイクロコントローラ、マイクロプロセッサ、デジタル信号プロセッサ、または他の任意のタイプのデータ処理システムであることが可能である。さらに、データ処理システム10は、任意のタイプの電気回路を使用して実装することが可能である。外部デバイス2は、メモリまたは任意のタイプの周辺デバイスを含め、あらゆるタイプの電気回路であることが可能である。代替の実施形態は、より多い、より少ない、または異なる外部集積回路群を含むことが可能である。さらに、バス4およびバス6は、任意の数のビットを使用して実装することが可能である。

【0008】

動作の際、システム集積回路22を使用して、プロセッサ12と外部デバイス2の間の通信を可能にする。つまり、プロセッサ12は、内部バス14および13を介してデータおよびアドレス情報をシステム集積回路22に送り、システム集積回路22は、データおよびアドレス情報を、外部デバイス2に適切な方法およびフォーマットで、バス4およびバス6を介して送る。プロセッサ12について、図2を参照して以下により詳細に説明する。

【0009】

図2は、本発明の一実施形態によるプロセッサ12の一部分を示している。プロセッサ12は、演算論理装置（ALU）24、アドレス・ジェネレータ26、命令パイプライン28、命令復号化回路30、レジスタ・ファイル・セット32、およびベクトル・オフセット・ジェネレータ39を含む。レジスタ・ファイル32は、コンテキスト0 34、コ

10

20

30

40

50

ンテキスト 1 3 5、およびコンテキスト N 3 6 のような複数のレジスタ・コンテキストを含む。したがって、レジスタ・ファイル 3 2 は、N + 1 個のレジスタ・コンテキストを含み、3 つだけを図 2 に示しているが、プロセッサ 1 2 は、いくつかのレジスタ・コンテキストをハードウェアがサポートすることが可能であるかに依存して、任意の数のレジスタ・コンテキストを含むことが可能である。また、レジスタ・ファイル 3 2 は、制御レジスタ・ファイル 3 8 も含む。内部アドレス・バス 1 4 がアドレス・ジェネレータ 2 6 に結合され、アドレス・ジェネレータ 2 6 は、レジスタ・ソース・バス 4 0 を介してレジスタ・ファイル 3 2 に結合され、内部データ・バス 1 3 にも結合される。ベクトル・オフセット・ジェネレータ 3 9 は、ベクトル・オフセット・バス 2 7 を介してアドレス・ジェネレータ 2 6 に結合される。内部データ・バス 1 3 は、命令パイプライン 2 8、A L U 2 4、およびレジスタ・ファイル 3 2 に結合される。命令復号化回路 3 0 は、命令バス 2 9 を介して命令パイプライン 2 8 に双方向で結合される。レジスタ・ファイル 3 2 は、フィールド情報バス 4 2 を介して A L U 2 4 に結合される。

#### 【 0 0 1 0 】

図 3 は、本発明の一実施形態によるレジスタ・コンテキスト 5 1 を示している。図 3 のレジスタ・コンテキスト 5 1 は、図 2 のコンテキスト 0 ~ N のいずれか 1 つを表すことが可能である。図 3 の実施形態では、レジスタ・コンテキスト 5 1 は、3 2 個の汎用レジスタ ( G P R ) 5 0、リンク・レジスタ 5 4、カウント・レジスタ 5 6、条件レジスタ 5 8、整数例外レジスタ 6 0、マシン状態レジスタ 6 2、およびコンテキスト制御レジスタ 6 4 を含む。リンク・レジスタ 5 4 は、サブルーチンを呼び出す際、およびサブルーチンから戻る際、サブルーチン・リンケージ情報を保持するのに使用される。カウント・レジスタ 5 6 は、命令のカウントされるループを処理するためのカウント情報を保持するのに使用される。条件レジスタ 5 8 は、条件コード計算の結果を保持するのに使用される。整数例外レジスタ 6 0 は、様々な例外ステータスを提供するのに使用される。マシン状態レジスタ 6 2 は、プロセッサ 1 2 内部の様々な機能のステータスを制御し、それらの機能のステータスを提供するのに使用される。コンテキスト制御レジスタ 6 4 は、以下により詳細に説明するとおり、本発明の一実施形態に従ってコンテキスト切り替えを提供するのに使用される。G P R 5 0 の 1 つは、現在のスタック・ポインタを格納するために確保されたスタック・ポインタ・レジスタ 5 2 であることに留意されたい。

#### 【 0 0 1 1 】

したがって、レジスタ・コンテキストとは、前述したレジスタ群 ( コンテキスト 5 1 のレジスタ群 ) の内容を指す。代替の実施形態は、レジスタ・コンテキスト 5 1 の同一のレジスタ群のすべて、または一部を有するものとしてレジスタ・コンテキストを定義するか、またはレジスタ・コンテキスト 5 1 のレジスタ群とは異なるレジスタ・セットを含むことが可能である。したがって、本明細書で使用するレジスタ・コンテキストは、任意の数の任意のタイプのレジスタを有するように定義することが可能である。通常、レジスタ・コンテキストは、プロセッサに関するプログラムのレジスタ・モデルの全部または一部を形成するレジスタ・リソース群を含む。通常の動作中、または電源投入またはリセットの後、データ処理システム 1 0 は、デフォルトでコンテキスト 0 3 4 を利用することが可能である。( 代替の実施形態では、通常の動作は、デフォルトで異なるコンテキストになることに留意されたい。 ) しかし、割り込みまたはプロセス切り替えが生じた場合、コンテキスト 0 3 4 内の値を壊さないように、データ処理システム 1 0 は、その割り込みを処理するため、またはその新たなプロセスまたはスレッドを実行するために、新たなコンテキストを ( コンテキスト 1 ないし N から ) 選択する。したがって、割り込み処理およびプロセス切り替え ( 例えば、マルチスレッド ) により、データ処理システム 1 0 内部でレジスタ・コンテキストを切り替える必要が生じる可能性がある。また、一部の実施形態では、レジスタ・コンテキスト内のレジスタ群の一部を別のレジスタ・コンテキストと共有することが望ましい可能性がある。したがって、以下に説明するとおり、レジスタ・コンテキストの一部分を別のレジスタ・コンテキストにマップして、オーバーヘッドを減らし、コンテキスト切り替え中の速度を高めるのに役立てることが可能である。

## 【 0 0 1 2 】

データ処理システム 10 では、例外および割り込みは、命令パイプライン 28 の復号化段階または実行段階において認識される。このため、命令が命令復号化回路 30 に与えられ、復号化された場合、通常の命令処理の代りに、割り込みが認識され、処理されることが可能である。本明細書で説明する一実施形態では、所与の割り込みが他の何らかの割り込みに優先されるかどうかを決める複数の割り込みレベルが存在する。このため、高い優先順位を有する割り込みは、より低い優先順位を有する割り込みより迅速に処理され、より低い優先順位を有する割り込みは、処理を待たなければならない。したがって、同一の優先順位を有する各割り込み、あるいは同一の優先順位を有する各割り込みまたは各タイプの割り込みは、所望される場合、同一のレジスタ・コンテキストを共有することが可能である。

10

## 【 0 0 1 3 】

割り込みが受け取られた場合、データ処理システム 10 は、例外処理シーケンスを実行することを始める。このシーケンス中、ベクトル・オフセット・ジェネレータ 39 は、ベクトル・オフセット・バス 27 を介してベクトル・オフセット値をアドレス・ジェネレータ 26 に与える。アドレス・ジェネレータ 26 は、ベクトル・オフセット値を使用して、割り込みを処理するために実行が開始されるべき命令アドレスを形成する。一実施形態では、ベクトル・オフセット値に加えて、ベクトル・オフセット・ジェネレータは、割り込み処理のために使用されるべきレジスタ・コンテキストを示すコンテキスト・セクタも与える。一実施形態では、コンテキスト・セクタは、ベクトル・オフセット値の一部分であるか、またはベクトル・オフセット・ジェネレータ 39 によって与えられる別個の値であることが可能である。また、コンテキスト・セクタは、レジスタ・ファイル 32 に直接に与えることも可能である。代替の実施形態では、コンテキスト・セクタは、メモリ（図示せず）から読み取られる値であることが可能であり、あるいは命令を介して受け取られることが可能である。データ処理システム 10 がレジスタ・ファイル 32 の中に 8 つのレジスタ・コンテキストを有するケースでは、コンテキスト・セクタは、レジスタ・コンテキストの 1 つを識別するのに使用される 3 ビットの値であることが可能である。

20

## 【 0 0 1 4 】

また、データ処理システム 10 は、1 つのプロセスから別のプロセス（それぞれのプロセスは異なるレジスタ・コンテキストで動作することが可能である）に、プロセッサ 12 が切り替えることが可能である場合、プロセス切り替えを行うことが可能である。例えば、マルチスレッド・アプリケーションでは、プロセッサ 12 は、異なる処理スレッド（または処理スレッドのグループ）が異なるレジスタ・コンテキストを使用する場合、様々な処理スレッドの間で絶えず切り替えを行うことが可能である。プロセス切り替えのケースでは、割り込みを使用して、プロセス切り替えをデータ処理システム 10 に示すことが可能である（割り込み処理が切り替えプロセスを含む場合）。あるいは、他の方法を使用して、プロセス切り替えが必要であることをアドレス・ジェネレータ 26 に示し、アドレス・ジェネレータ 26 が新たなプロセスの開始アドレスを生成することが可能である。また、プロセス切り替え後、新たなプロセスにいずれのレジスタ・コンテキストが必要とされるかを示すようにコンテキスト・セクタも与えられる。前述したとおり、コンテキスト・セクタは、様々な異なる形で（すなわち、ベクトル・オフセット・ジェネレータ 39 から、メモリから、ユーザ命令からなど）与えることが可能であり、正しいレジスタ・コンテキストが選択されることが可能であるように、レジスタ・ファイル 32 に直接に、または間接的に（例えば、アドレス・ジェネレータ 26 を介して）与えることが可能である。

30

40

## 【 0 0 1 5 】

レジスタ・コンテキストが確立されると、プロセッサ 12 によって実行される命令は、現在、確立されているコンテキストに対応する適切な汎用レジスタ群（GPR 群 50）または専用レジスタ群（例えば、LR 54、CTR 56、CR 58、XER 60、MSR 62、または CTXCR 64）を参照する。他のコンテキスト内のレジスタ群は、影響を受

50

けず（以下に説明するとおり、マッピングが確立されていない限り）、このため、現在、確立されているコンテキストに関する命令の実行に先立って、メモリに代替のコンテキストを保存すること、または復元することを実行する必要は全くない。これにより、オーバーヘッドの節減が可能になる。

#### 【 0 0 1 6 】

図 4 および図 5 は、データ処理システム 10 内部で使うことが可能なレジスタ・コンテキスト内の典型的なマッピングを示している。図 4 は、3 つのレジスタ・コンテキスト、すなわち、コンテキスト 0 70、コンテキスト 1 72、およびコンテキスト 2 74 を示している。以上のレジスタ・コンテキストは、図 2 のコンテキスト 0 ~ N の範囲内のコンテキストの 3 つを表すことが可能である。図 4 の例では、コンテキスト 0 70 は、データ処理システム 10 の通常の動作に対応し、コンテキスト 1 72 は、重要な割り込み（最高優先順位）に対応し、コンテキスト 2 74 は、外部割り込み（低い優先順位）に対応するものと想定する。前述したとおり、一部のケースでは、複数のレジスタ・コンテキストがレジスタ群の一部を「共有する」ことが望ましい。したがって、図 4 の例では、矢印 82 で示すとおり、レジスタ・コンテキスト 2 74 のスタック・ポインタ・レジスタ 80 が、レジスタ・コンテキスト 1 72 のスタック・ポインタ 78 にマップされ、レジスタ・コンテキスト 1 72 とレジスタ・コンテキスト 2 74 が同一のスタック・ポインタを共有することが可能であり、両方のレジスタ・コンテキストにおいて同一のスタック・ポインタ値が使用されるようになっていることを示している。このマッピングは、オーバーヘッドを減らし、スタック・ポインタの整合性を保つのに役立つ。したがって、外部割り込みを処理するときには、コンテキスト 2 74 がデータ処理システム 10 によって選択される。しかし、スタック・ポインタ・レジスタ 80 がスタック・ポインタ・レジスタ 78 にマップされているので、レジスタ・コンテキスト 1 72 内のスタック・ポインタ・レジスタ 78 が、そのスタック・ポインタにアクセスするためにレジスタ・コンテキスト 2 74 における動作中にアクセスされる。つまり、レジスタ・コンテキスト 2 74 を選択する現在のコンテキスト値で動作している間、スタック・ポインタ・レジスタ 80 にアクセスしようと試みる命令およびその他の動作は、レジスタ・コンテキスト 1 72 内のスタック・ポインタ・レジスタ 78 にアクセスするように向けかえられる。これにより、別々のスタック・ポインタ・レジスタ 80 と 78 を同期させるオーバーヘッドなしに、コンテキスト 1 とコンテキスト 2 の間で単一の矛盾しないスタックおよびスタック・ポインタ値が共有されることが可能になる。

#### 【 0 0 1 7 】

スタック・ポインタ・レジスタ 76（コンテキスト 0 70 の）とスタック・ポインタ・レジスタ 78（コンテキスト 1 72 の）はマップされていないことに留意されたい。したがって、これらのレジスタ・コンテキストで動作している場合、スタック・ポインタにアクセスする際に他のいずれのレジスタ・コンテキストにもアクセスする必要がある。レジスタ・コンテキスト 70、72、および 74 のそれぞれの中のコンテキスト制御レジスタ 77、79、および 75 はそれぞれ、対応するレジスタ・コンテキストのスタック・ポインタがマップされているかどうかを示し、マップされている場合、他のいずれのレジスタ・コンテキストにマップされているかを示す。コンテキスト制御レジスタの詳細を、図 6 を参照して以下により詳細に説明する。

#### 【 0 0 1 8 】

図 5 は、別の例による 3 つのレジスタ・コンテキスト、すなわち、レジスタ・コンテキスト 1 90、レジスタ・コンテキスト 2 92、およびレジスタ・コンテキスト 3 94 を示している。図 4 の場合と同様に、図 5 のレジスタ・コンテキストは、図 2 のレジスタ・コンテキスト 0 ~ N のレジスタ・コンテキストの 3 つを表すことが可能である。図 5 の例では、レジスタ・コンテキスト 1 90 は、プロセス A に対応し、レジスタ・コンテキスト 2 92 は、プロセス B に対応し、レジスタ・コンテキスト 3 94 は、プロセス C に対応する。したがって、データ処理システム 10 は、プロセス A を実行している場合、レジスタ・コンテキスト 1 90 で動作する。プロセス切り替え時（例えばプロセス A

10

20

30

40

50

からプロセスBへの切り替え)、コンテキスト・セクタにより、プロセスBを実行する際に使用するためのレジスタ・コンテキスト2 92が選択される。図4に関連して前述したとおり、スタック・ポインタ・レジスタのそれぞれは、異なるレジスタ・コンテキストにマップされる能力を有する。例えば、図5で、レジスタ・コンテキスト1 90のスタック・ポインタ・レジスタ96が、矢印124で示すとおり、レジスタ・コンテキスト2 92のスタック・ポインタ・レジスタ98にマップされる。したがって、プロセスAを実行している(レジスタ・コンテキスト1 90を使用して)場合、スタック・ポインタへのアクセスにより、実際には、異なるレジスタ・コンテキスト(すなわち、レジスタ・コンテキスト2 92)内のスタック・ポインタ・レジスタ98へのアクセスがもたらされる。ただし、レジスタ・コンテキスト3 94のスタック・ポインタ・レジスタ100はマップされていないことに留意されたい。また、一実施形態では、階層型マッピングを有することも可能である。例えば、スタック・ポインタ・レジスタ96がスタック・ポインタ・レジスタ98にマップされるのと同様に、スタック・ポインタ・レジスタ98も、例えば、スタック・ポインタ・レジスタ100にマップされることが可能である。また、特定のスタック・ポインタ・レジスタに、複数のスタック・ポインタ・レジスタがマップされることも可能である。例えば、スタック・ポインタ・レジスタ98に、スタック・ポインタ・レジスタ100とスタック・ポインタ・レジスタ96がマップされることが可能である。その他のマッピングも同様に可能である。

#### 【0019】

図5のレジスタ・コンテキスト群は、レジスタのグループ化も含む。例えば、汎用レジスタ群は、4つのレジスタのグループにグループ化される。レジスタ・コンテキスト1 90において、GPR4~7が一緒にグループ化されてレジスタ・グループ102になり、GPR8~11が一緒にグループ化されてレジスタ・グループ104になり、GPR28~31がグループ化されてレジスタ・グループ106になる。したがって、レジスタ・コンテキスト1 90は、図5の例では、それぞれ4つのレジスタの3つのグループ(グループ102、104、および106)を含み、これらのグループのそれぞれが、異なるレジスタ・コンテキストにマップされる(グループとして)ことが可能である。代替の実施形態では、任意の数およびタイプのレジスタをグループ化することが可能である。あるいは、所望される細分性に依存して、それぞれの個別のレジスタを別個のグループと見なすことが可能である。同様に、レジスタ・コンテキスト2 92は、4つのレジスタの3つのグループ、すなわち、GPR4~7を有するグループ114、GPR8~11を有するグループ116、およびGPR28~31を有するグループ118)を含む。また、レジスタ・コンテキスト3 94も、4つのレジスタの3つのグループ、すなわち、GPR4~7を有するグループ108、GPR8~11を有するグループ110、およびGPR28~31を有するグループ112)を含む。以上のグループ化により、レジスタのグループが、異なるレジスタ・コンテキストの間でマッピングされることが可能になる。

#### 【0020】

例えば、矢印120で示すとおり、レジスタ・コンテキスト2 92のグループ118が、レジスタ・コンテキスト1 90のグループ106にマップされる。矢印126で示すとおり、レジスタ・コンテキスト3 94のグループ112も、レジスタ・コンテキスト1 90のグループ106にマップされる。つまり、グループ106のレジスタ群は、3つすべてのレジスタ・コンテキスト、すなわち、レジスタ・コンテキスト1 90、レジスタ・コンテキスト2 92、およびレジスタ・コンテキスト3 94によって共有される。したがって、プロセスBまたはプロセスCを実行している場合、現在のレジスタ・コンテキスト(それぞれ、レジスタ・コンテキスト2 92またはレジスタ・コンテキスト3 94)のGPR28~31へのアクセスにより、実際には、レジスタ・コンテキスト1 90のGPR28~31へのアクセスがもたらされる。やはり図5に示されていることとして、矢印122で示すとおり、レジスタ・コンテキスト1 90のグループ104が、レジスタ・コンテキスト2 92のグループ116にマップされている。つまり、グループ116のレジスタ群は、レジスタ・コンテキスト1 90とレジスタ・コンテ

スト 2 9 2 の両方によって共有される。したがって、プロセス A を実行している場合、現在のレジスタ・コンテキストの G P R 8 ~ 1 1 へのアクセスにより、実際には、レジスタ・コンテキスト 2 9 2 の G P R 8 ~ 1 1 へのアクセスがもたらされる。したがって、単一のレジスタ（スタック・ポインタ・レジスタ 9 6、9 8、または 1 0 0 のような）のマッピングであるか、レジスタのグループのマッピングであるかに関わらず、任意の数のマッピングが存在することが可能である。また、各レジスタ・コンテキストは、あるレジスタ群を 1 つのレジスタ・コンテキストにマップしており、他のレジスタ群をもう 1 つのレジスタ・コンテキストにマップしていることも可能である。また、レジスタ、またはグループのレジスタに、複数のレジスタ・コンテキストのレジスタ群がマップされていることも可能である。

10

#### 【 0 0 2 1 】

各レジスタ・コンテキストのマッピングは、各レジスタ・コンテキストのコンテキスト制御レジスタ（例えば、図 5 のコンテキスト制御レジスタ 1 2 8、1 3 0、および 1 3 2）の中で定義される。したがって、図 2 の各レジスタ・コンテキスト 0 ~ N は、各レジスタ・コンテキストの中に含まれることが可能である（図 4 および 5 のように）か、または別々に（図 2 の制御レジスタ・ファイル 3 8 の中などに）格納されることが可能な、対応するコンテキスト制御レジスタを有する。図 6 は、本発明の一実施形態によるコンテキスト制御レジスタ 1 4 0 の内容を示している。コンテキスト制御レジスタ 1 4 0 は、図 4 のコンテキスト制御レジスタ 7 7、7 9、7 5、または図 5 のコンテキスト制御レジスタ 1 2 8、1 3 0、および 1 3 2 を指すことが可能である。一実施形態では、コンテキスト制御レジスタ 1 4 0 は、レジスタ群のマッピングを制御する様々な異なるフィールドを有し、現在のコンテキスト情報、代替のコンテキスト情報、および保存されたコンテキスト情報を保持する専用 3 2 ビット・レジスタである。

20

#### 【 0 0 2 2 】

コンテキスト制御レジスタ 1 4 0 のビット 0 は、複数のレジスタ・コンテキストの使用を有効にするコンテキスト・イネーブル・フィールド 1 4 2 に対応する。例えば、コンテキスト・イネーブル・フィールド 1 4 2 がネゲートされた場合、単一のコンテキストだけが有効にされ、コンテキスト制御 1 4 0 内の他のすべての制御フィールドは無視され、現在のコンテキストは、既定のレジスタ・コンテキスト（図 2 に示した実施形態では、レジスタ・コンテキスト 0 3 4 である）に設定される。コンテキスト・イネーブル・フィールド 1 4 2 がアサートされた場合、複数のコンテキストが有効になる。ビット 3 ~ 5 は、ハードウェアによってサポートされる最大コンテキスト数を示す読み取り専用フィールドであるコンテキスト数フィールド 1 4 4 に対応する。図 6 の例では、0 0 0 という値により、1 つのコンテキストがサポートされることが示され、1 1 1 という値により、8 つのレジスタ・コンテキストがハードウェアによってサポートされることが示される。データ処理システム 1 0 が、8 つより多くのレジスタ・コンテキストをサポートすることが可能な場合、コンテキスト数フィールド 1 4 4 のために追加のビットを使用することが可能である。ただし、図 6 の実施形態では、最大で 8 つのレジスタ・コンテキストがサポートされるものと想定する。

30

#### 【 0 0 2 3 】

ビット 6 ~ 8 は、現在、有効にされているレジスタ・コンテキストを定義する現行コンテキスト・フィールド 1 4 6 に対応する。一実施形態では、このフィールドは、リセット時にクリアされて 0 になり、既定のレジスタ・コンテキストはレジスタ・コンテキスト 0 であることを示す。現行コンテキスト・フィールド 1 4 6 は、図 2 のベクトル・オフセット・ジェネレータ 3 9 によって与えられるなど、様々な異なる形で与えられることが可能な前述したコンテキスト・セレクトに対応する。したがって、コンテキスト切り替え（割り込み、またはプロセス切り替えによって生じさせられる）時に、現行コンテキスト・フィールド 1 4 6 は、コンテキスト・セレクトによって示される新たなレジスタ・コンテキストに設定される。例えば、図 5 を参照すると、データ処理システム 1 0 が現在、プロセス A において実行されている場合、プロセス B へのコンテキスト切り替え時に、コンテキ

40

50



スト・セクタによりレジスタ・コンテキスト 2 が示され、2 という値が、レジスタ・コンテキスト 2 9 2 のコンテキスト制御レジスタの現行コンテキスト・フィールドに書き込まれる。

#### 【 0 0 2 4 】

各レジスタ・コンテキストは、独自のコンテキスト制御レジスタを有し、フィールドのいくつかは、異なるコンテキスト制御レジスタの間で共有されることが可能であることに留意されたい。例えば、単一のコンテキスト・イネーブル・ビット、単一のコンテキスト数フィールド、および単一の現行コンテキスト・フィールドが実装されることが可能であり、すべてのコンテキスト制御レジスタによって使用される。というのは、値は、異なるコンテキスト制御レジスタの間で常に同一だからである。代替の実施形態は、各コンテキスト制御レジスタに関してコンテキスト・イネーブル・フィールドまたはコンテキスト数フィールド、または現行コンテキスト・フィールドを使用することも可能であるが、それぞれに関して単一の共有フィールドを使用することにより、ハードウェア要件が少なくなる。

#### 【 0 0 2 5 】

ビット 9 ~ 1 1 は、前に有効にされたコンテキストを定義する保存済みコンテキスト・フィールド 1 4 8 に対応する。このフィールドも、リセット時にクリアされて 0 になることが可能であることに留意されたい。したがって、プロセス A からプロセス B に切り替える前述の例では (図 5 参照)、現行コンテキスト・フィールドは、2 (レジスタ・コンテキスト 2 9 2 を表す) に設定され、保存済みコンテキスト・フィールドは、1 (前のコンテキスト、レジスタ・コンテキスト 1 9 0 を表す) に設定される。

#### 【 0 0 2 6 】

ビット 1 2 ~ 1 4 は、レジスタ・グループに関するコンテキスト・マッピングを定義するのに使用される代替で有効にされるコンテキストを定義する代替コンテキスト・フィールド 1 5 0 に対応する。ビット 1 5 ~ 1 8 は、マッピング・フィールド 1 5 1 に対応する。ビット 1 5 は、レジスタ・グループ A ( G P R 4 ~ 7 と定義される ) に対応し、ビット 1 6 は、レジスタ・グループ B ( G P R 8 ~ 1 1 と定義される ) に対応し、ビット 1 7 は、レジスタ・グループ C ( G P R 1 6 ~ 2 3 と定義される ) に対応し、ビット 1 8 は、レジスタ・グループ D ( G P R 2 7 ~ 3 1 と定義される ) に対応する。レジスタ・グループ A ~ D のそれぞれは、対応するビットをアサートすることによって独立に有効にすることが可能である。例えば、ビット 1 5 がアサートされた場合、グループ A が有効にされて、グループ A が代替コンテキスト・フィールドによって定義されたレジスタ・コンテキストにマップされるようになる。ビット 1 5 がネゲートされた場合、グループ A はマップされない。同様に、ビット 1 6、ビット 1 7、またはビット 1 8 がアサートされた場合、レジスタの対応するグループ (それぞれ B、C、または D) が、代替コンテキスト・フィールドによって定義されたレジスタ・コンテキストにマップされる。ビット 1 6、ビット 1 7、またはビット 1 8 がネゲートされた場合、レジスタの対応するグループ (それぞれ B、C、または D) はマップされない。したがって、図 5 を参照すると、レジスタ・コンテキスト 1 9 0 のコンテキスト制御レジスタが、代替コンテキスト・フィールドの中に 2 を含み、レジスタの選択された 1 つまたは複数のグループがレジスタ・コンテキスト 2 9 2 にマップされることを示している。また、ビット 1 6 ( G P R 8 ~ 1 1 を有するグループ B に対応する ) がアサートされて、レジスタ・コンテキスト 1 9 0 のグループ 1 0 4 がレジスタ・コンテキスト 2 9 2 のグループ 1 1 6 にマップされるようになる。

#### 【 0 0 2 7 】

図 6 の典型的なコンテキスト制御レジスタである、コンテキスト制御レジスタ 1 4 0 では、単一の代替コンテキスト・フィールドが利用可能であり、各グループ ( A ~ D ) がその同じ代替レジスタ・コンテキストにマップされるように有効にされることが可能である。つまり、グループ A が特定のレジスタ・コンテキストにマップされた場合、グループ C ~ D は、その同じコンテキストにだけマップすることが可能である。ただし、代替の実施形態では、レジスタのそれぞれの別個のグループ (グループ A ~ D のような) が、対応す

10

20

30

40

50

る代替コンテキスト・フィールドを有して、異なる代替レジスタ・コンテキストにマップされることが可能であるようになっていてもよい。あるいは、グループのグループに関して別個の代替コンテキスト・フィールド（例えば、グループAおよびグループBに関して1つの代替コンテキスト・フィールド、グループCおよびグループDに関して別の代替コンテキスト・フィールド）を使用することが可能である。また、グループは、任意の形で定義することが可能である。例えば、各グループは、4つのより多い、または少ないレジスタを有することが可能であり、各グループが単一のレジスタであること、または各グループが異なる数のレジスタを有することが可能である。また、代替の実施形態では、より多い、またはより少ないグループをより多い、またはより少ない代替コンテキスト・フィールドとともに使用することが可能である。したがって、図6のコンテキスト制御レジスタ140は、一例に過ぎない。また、各フィールドは、必要に応じて、より多い、またはより少ないビットを使用してフィールドの値を表すことも可能である。コンテキスト制御140は、未使用のビット1、2、19~23、および28~31を含むが、代替の実施形態は、未使用のビットを全く含まないこと、またはコンテキスト制御情報を格納するのに複数のレジスタを要することも可能である。

#### 【0028】

コンテキスト制御レジスタ140のビット24は、図4と図5の両方に関連して説明したとおり、スタック・ポインタのマッピングを有効にするスタック・ポインタ・コンテキスト・イネーブル・フィールド152に対応する。ビット25~27は、スタック・ポインタに関する代替レジスタ・コンテキストを選択するスタック・ポインタ・コンテキスト選択フィールド154に対応する。したがって、ビット24がアサートされた場合、スタック・ポインタは、スタック・ポインタ・コンテキスト選択フィールド154によって示されるレジスタ・コンテキストにマップされるが、ビット24がネゲートされた場合、スタック・ポインタはマップされない（すなわち、現行コンテキスト・フィールド146によって定義された現行コンテキストに留まる）。スタック・ポインタ・コンテキスト選択フィールド154は、データ処理システム10内部の8つのレジスタ・コンテキストのいずれがスタック・ポインタに関する代替のコンテキストとして使用されるべきかを示すことが可能な3ビットの値である。例えば、000という値が、レジスタ・コンテキスト0に対応し、111がレジスタ・コンテキスト7に対応することが可能である。したがって、スタック・ポインタ・コンテキスト選択フィールド154が001に設定されている場合、（かつスタック・ポインタ・コンテキスト・イネーブル・フィールド152がアサートされた場合）、現在のコンテキストのスタック・ポインタは、レジスタ・コンテキスト1にマップされる。例えば、図4を再び参照すると、レジスタ・コンテキスト274のコンテキスト制御レジスタのスタック・ポインタ・コンテキスト・イネーブル・フィールドがアサートされ、スタック・ポインタ・コンテキスト選択フィールドが001に設定されて、スタック・ポインタ・レジスタ80がレジスタ・コンテキスト172のスタック・ポインタ・レジスタ78にマップされることを示す。また、代替の実施形態は、スタック・ポインタの共有を可能にするだけではない他の個々のレジスタのマッピングも可能にするフィールドを含むことが可能であることにも留意されたい。

#### 【0029】

コンテキスト制御レジスタは、様々な異なる形でプログラミングすることが可能である。例えば、一実施形態では、各コンテキスト制御レジスタを、直接にユーザによってプログラミングすることが可能である。あるいは、コンテキスト制御レジスタは、現在のコンテキスト制御レジスタの代替のコンテキスト・フィールドを使用して間接的にマップされることが可能である。例えば、一実施形態では、電源投入またはリセットの後、データ処理システムは、デフォルトでレジスタ・コンテキスト0になる。次に、代替コンテキスト・フィールドが、いずれのレジスタ・コンテキストのコンテキスト制御レジスタがプログラミングされるかを示す値に設定されることが可能である。例えば、レジスタ・コンテキスト0になっている間、レジスタ・コンテキスト0のコンテキスト制御レジスタの代替コンテキスト・フィールドに2という値を書き込むことにより、専用レジスタを介してレジ

10

20

30

40

50

スタ・コンテキスト 2 のコンテキスト制御レジスタのプログラミングへのアクセスが可能になる。コンテキスト制御フィールドのすべてがプログラミングされた後、それらのフィールドがすべて同時に有効にされる（単一の共有コンテキスト・イネーブル・フィールドを有するケースでは、そのビットをアサートすることによって行われる）ことが可能である。また、一実施形態では、コンテキスト制御レジスタ群のプログラミング中、割り込み処理をオフにすることが可能である。

#### 【 0 0 3 0 】

コンテキスト制御レジスタ 1 4 0 は、特定のフィールドおよびビットの位置に関連して説明してきたことに留意されたい。代替の実施形態は、必要に応じて、より多い、またはより少ないフィールドを含むことが可能であり、各フィールドは、必要に応じて、より多い、またはより少ないビットを含むことが可能であることに留意されたい。また、代替の実施形態では、コンテキスト制御レジスタ群は、データ処理システム 1 0 内部のいずれの場所に配置してもよいし、あるいはデータ処理システム 1 0 の外部に配置してもよい。

#### 【 0 0 3 1 】

かくして、オーバーヘッドのより少ない、柔軟性のあるコンテキスト選択を提供するために、どのようにコンテキスト制御レジスタを使用可能であることを理解することが可能であろう。データ処理システム 1 0 内部のコンテキスト切り替え時に、新たなレジスタ・コンテキストのコンテキスト制御レジスタが更新される。例えば、新たなレジスタ・コンテキストは、現行コンテキスト・フィールドに書き込まれ、前のコンテキストは、保存済みコンテキスト・フィールドに書き込まれ、新たなレジスタ・コンテキスト内で動作する際に、マッピング・フィールドの中で与えられるレジスタ・マッピングが使用される。レジスタ・マッピングにより、異なるレジスタ・コンテキストがレジスタ値を共有することが可能になる。また、レジスタ・マッピングにより、現在のレジスタ・コンテキストのコンテキスト制御レジスタによって定義される現在のレジスタ・コンテキスト外部の他のレジスタ・コンテキスト群のアクセスも可能になる。また、ユーザがプログラミング可能なコンテキスト制御レジスタ群により、どのようにマッピングを定義するかについて柔軟性が可能になる。したがって、本明細書で説明する本発明の一態様により、代替レジスタ・コンテキストの一部分を現在のレジスタ・コンテキストにマップする（またその逆を行う）柔軟性のある機構が提供され、複数のコンテキストの間で共通のスタック・ポインタの柔軟性のある共有が可能になり、リアルタイム・パフォーマンスの向上がもたらされる。現在のコンテキストの一部分を代替のコンテキストにマップすることにより、動作コンテキスト間で情報値を転送することのオーバーヘッドを除く、またはなくすることが可能になり、パフォーマンスの向上および柔軟性がもたらされる。

#### 【 0 0 3 2 】

以上の明細では、本発明を特定の実施形態に関連して説明した。しかし、添付の特許請求の範囲に記載する本発明の範囲を逸脱することなく、様々な改変および変更を行うことが可能であることが当業者には認められよう。例えば、ブロック図は、例示したブロックとは異なるブロックを有することが可能であり、より多い、またはより少ないブロックを有すること、または異なる形で構成されることも可能である。したがって、本明細書および図は、限定するものではなく、例示するものと考えられるべきであり、すべてのそのような改変形態が本発明の範囲に含まれるものとする。

#### 【 0 0 3 3 】

特定の実施形態に関して、利点、その他の優位点、および問題解決法を説明してきた。しかし、これらの利点、優位点、問題解決法、ならびに何らかの利点、優位点、または問題解決法を生じさせる、またはより顕著し得る要素は、一部または全部の請求項の不可欠な、必要な、または本質的な特徴または要素と解釈すべきではない。本明細書で使用する「からなる (comprises)」、「comprising (からなる)」、またはこの用語の他のあらゆる変化形は、排他的でない包含を範囲に含むものとし、したがって、要素のリストからなるプロセス、方法、物品、または装置は、それらの要素を含むだけでなく、明確にリストアップされていない、またはそのようなプロセス、方法、物品、ま

10

20

30

40

50

たは装置に固有でない他の要素も含み得る。

【図面の簡単な説明】

【 0 0 3 4 】

【図 1】本発明の一実施形態によるデータ処理システムを示すブロック図。

【図 2】本発明の一実施形態によるプロセッサを示すブロック図。

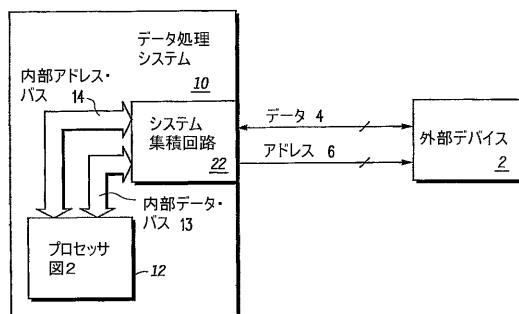
【図 3】本発明の一実施形態によるレジスタ・コンテキストを示す図。

【図４】本発明の様々な実施形態によるレジスタ・コンテキスト内の典型的なマッピングを示す図。

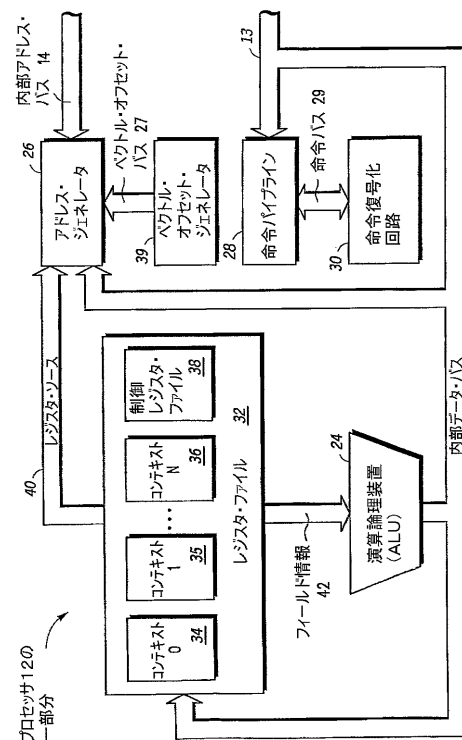
【図 5】本発明の様々な実施形態によるレジスタ・コンテキスト内の典型的なマッピングを示す図。

10

【圖 1】



【圖 2】





---

フロントページの続き

(72)発明者 モイヤー、ウィリアム シー .  
アメリカ合衆国 7 8 6 2 0 テキサス州 ドリッピング スプリングス ピア ブランチ ロー  
ド 1 0 0 5

(72)発明者 アレンズ、ジョン エイチ .  
アメリカ合衆国 7 8 7 5 0 テキサス州 オースティン ジャンシー ドライブ 8 3 1 3

審査官 井上 宏一

(56)参考文献 特表 2 0 0 2 - 5 1 2 3 9 9 ( J P , A )  
特開平 0 4 - 0 7 6 6 4 5 ( J P , A )  
特開 2 0 0 0 - 2 4 2 5 0 5 ( J P , A )  
特開平 0 4 - 2 4 2 4 3 3 ( J P , A )  
特開平 0 2 - 0 1 4 3 3 0 ( J P , A )  
特開平 1 1 - 2 8 8 3 7 2 ( J P , A )

(58)調査した分野(Int.Cl. , D B 名)  
G06F 9/46 -9/54